Project on Group LASSO Problem

Welkin507

2023 年 12 月 10 日

目录

1 Problem Description

Consider the group LASSO problem

$$\min_{x\in\mathbb{R}^{n\times l}} \frac{1}{2}\|Ax-b\|_F^2 + \mu\|x\|_{1,2}$$

where the data $A \in \mathbb{R}^{m\times n}, b \in \mathbb{R}^{m\times l}$ and $\mu > 0$ are given, and

$$\|x\|_{1,2} = \sum_{i=1}^{n} \|x(i,1:l)\|_2.$$

Note that $x(i,1:l)$ is the i-th row of the matrix x. Here, both x and b are matrices but they are written in small letters for the convenience of coding. The test data are generated as follows:

```
seed = 97006855;
ss = RandStream('mt19937ar','Seed',seed);
RandStream.setGlobalStream(ss);
n = 512;
m = 256;
A = randn(m,n);
k = round(n*0.1); l = 2;
A = randn(m,n);
p = randperm(n); p = p(1:k);
u = zeros(n,l); u(p,:) = randn(k,l);
b = A*u;
mu = 1e-2;
```

1. Solve (1.1) using CVX by calling different solvers Mosek and Gurobi.
2. First write down an equivalent model of (1.1) which can be solved by calling mosek and gurobi directly, then implement the codes.
3. First write down, then implement the following algorithms in Matlab (or Python) implement the codes.

The report and code refer to the following website and utilize ChatGPT and Copilot, which helps a lot.

`http://faculty.bicmr.pku.edu.cn/~wenzw/optbook/pages/contents/contents.html`

2 Use CVX by calling MOSEK and Gurobi

2.1 CVX

CVX是一个求解优化问题的模型语言，可以调用不同的求解器来求解优化问题。CVX的安装和使用可以参考http://cvxr.com/cvx/。

这里，我们使用

```
cvx_begin
    variable x(n, l)
    minimize(square_pos(norm(A * x - b, 'fro')) + mu * sum(norms(x, 2, 2)))
cvx_end
```

描述问题，调用求解器求解。求解器可以通过设置cvx_solver来指定，这里我们使用MOSEK和Gurobi求解器。

2.2 Experiments and Interpretation

Method	CPU (s)	Iterations	Optimal Value	Sparsity	Err. to Exact	Err. to CVX-Mosek	Err. to CVX-Gurobi
CVX-Mosek	2.66	11	5.80563×10^{-1}	0.103	1.90×10^{-5}	0.00×10^{0}	4.74×10^{-7}
CVX-Gurobi	0.72	25	5.80563×10^{-1}	0.103	1.87×10^{-5}	4.74×10^{-7}	0.00×10^{0}

表 1: CVX-Mosek and CVX-Gurobi.

从表格中可以看出，CVX-Mosek和CVX-Gurobi的结果极其接近，且与精确解的误差都很小。两者迭代次数都很少，CVX-Mosek迭代次数更少但是的运行时间要长一些。

3 Equivalent Model and Implement

3.1 Model Transformation

我们先将此问题转化为一个二次规划问题(SOCP)。引入变量s, t_i, u，则有

$$\begin{aligned} \min_{x\in\mathbb{R}^{m\times n}, s\in\mathbb{R}, t\in\mathbb{R}^n} \quad & \tfrac{1}{2}s + \mu\textstyle\sum_{i=1}^{n} t_i \\ \text{s.t.} \quad & t_i \geq \| x(i,:) \|_2,\ i = 1, ..., m, \\ & u = 1, \\ & us \geq \| Ax - b \|_{\mathrm{F}}^2 . \end{aligned}$$

也可以将这个问题转化为二次约束二次规划(QCQP)，引入变量y_{ij}和t_i，则有

$$\begin{aligned} \min_{x\in\mathbb{R}^{m\times n},s\in\mathbb{R},t\in\mathbb{R}^n} \quad & \tfrac{1}{2}\|y\|_F^2+\mu\textstyle\sum_{i=1}^n t_i \\ \text{s.t.} \quad & t_i^2 \geq \| x(i,:)\|_2^2,\ i=1,...,n, \\ & Ax-b-y=0, \\ & t_i\geq 0, i=1,...,n. \end{aligned}$$

在MATLAB中直接调用Mosek和Gurobi求解器可以分别求解这两个问题。

3.2 Implement in Mosek and Gurobi

实验结果如下

Method	CPU (s)	Iterations	Optimal Value	Sparsity	Err. to Exact	Err. to CVX-Mosek	Err. to CVX-Gurobi
CVX-Mosek	2.66	11	5.80563×10^{-1}	0.103	1.90×10^{-5}	0.00×10^{0}	4.74×10^{-7}
CVX-Gurobi	0.72	25	5.80563×10^{-1}	0.103	1.87×10^{-5}	4.74×10^{-7}	0.00×10^{0}
Mosek	1.29	10	5.80556×10^{-1}	0.103	3.75×10^{-5}	1.85×10^{-5}	1.88×10^{-5}
Gurobi	0.31	12	5.80556×10^{-1}	0.105	3.77×10^{-5}	1.87×10^{-5}	1.90×10^{-5}

表 2: Mosek and Gurobi.

可以看到，Mosek和Gurobi的结果与CVX-Mosek和CVX-Gurobi的结果基本一致，但是迭代次数更少，运行时间更短，解的稀疏程度没有明显变化。Gurobi仍然比Mosek快一些。

4 Algorithms

我们在MATLAB中实现了如下算法，以解决Group LASSO问题。

4.1 (a) Subgradient method for the primal problem

记

$$\psi(x)=\frac{1}{2}\|Ax-b\|_F^2+\mu\|x\|_{1,2}=f(x)+h(x)=f(x)+\sum_{i=1}^n h_i\left(x\right)$$

其中

$$f(x)=\frac{1}{2}\|Ax-b\|_F^2,\quad h_i(x)=\mu\|x(i,:)\|_2$$

注意到

$$\begin{aligned} \|A(x+tv)-b\|_F^2 &= \langle A(x+tv)-b, A(x+tv)-b\rangle \\ &= \langle Ax+Atv-b, Ax+Atv-b\rangle \\ &= \langle Ax-b, Ax-b\rangle+2\langle Ax-b, Atv\rangle+\langle Atv, Atv\rangle \\ &= \|Ax-b\|_F^2+2t\langle A^T(Ax-b), v\rangle+\mathcal{O}(t^2). \end{aligned}$$

因此

$$\nabla f(x) = A^T(Ax - b)$$

考虑2范数的次梯度:

$$\|x\|_2 = \sqrt{\sum_{i=1}^{n} x_i^2}$$

在 $x \neq 0$ 处可微:

$$\nabla \|x\|_2 = \frac{d}{dx}\sqrt{\sum_{i=1}^{n} x_i^2} = \frac{1}{2\sqrt{\sum_{i=1}^{n} x_i^2}} \cdot 2x = \frac{x}{\|x\|_2}$$

在原点处 $x = 0$, 根据定义，2-norm是满足如下要求的g:

$$\|y\|_2 - \|0\|_2 \geq \langle g, y - 0 \rangle$$

$\forall y \in \mathbb{R}^n$, 也就是:

$$\|y\|_2 \geq \langle g, y \rangle$$

容易根据Cauchy-Schwarz inequality, $|\langle g, y \rangle| \leq \|g\|_2 \|y\|_2$, 证明$\|g\|_2 \leq 1$是所有的次梯度。
因此，2范数 $\|x\|_2$ 的次梯度为:

$$\partial \|x\|_2 = \begin{cases} \{\frac{x}{\|x\|_2}\}, & \text{if } x \neq 0 \\ \{g \in \mathbb{R}^n : \|g\|_2 \leq 1\}, & \text{if } x = 0 \end{cases}$$

因此，我们有

$$\partial \mu \|x(i, 1:l)\|_2 = \begin{cases} \left\{ \mu \frac{x(i,1:l)}{\|x(i,1:l)\|_2} \right\}, & \text{if } x(i, 1:l) \neq 0 \\ \left\{ g \in \mathbb{R}^l : \|g\|_2 \leq \mu \right\}, & \text{if } x(i, 1:l) = 0 \end{cases}$$

$$\partial h(x) = \partial \mu \|x\|_{1,2} = \left\{ G \in \mathbb{R}^{n \times l} : G(i, 1:l) \in \partial \mu \|x(i, 1:l)\|_2 \text{ for each } i = 1, \ldots, n \right\}$$

也就是对于每一个 $i = 1, \ldots, n$，对每一行的2范数分别取次梯度，然后拼接起来。
次梯度法的迭代公式如下

$$x^{k+1} = x^k - \alpha_k g^k$$

代码实现的核心步骤为

```
sub_g = g + mu * x ./ repmat(max(1e-8, norms(x, 2, 2)), 1, size(x, 2));
x = x - alpha * sub_g;
```

在数值实验中，我们固定步长，并在每一步将接近0的x调整为0，最终实验结果为

Method	CPU (s)	Iterations	Optimal Value	Sparsity	Err. to Exact	Err. to CVX-Mosek	Err. to CVX-Gurobi
CVX-Mosek	2.66	11	5.80563×10^{-1}	0.103	1.90×10^{-5}	0.00×10^{0}	4.74×10^{-7}
CVX-Gurobi	0.72	25	5.80563×10^{-1}	0.103	1.87×10^{-5}	4.74×10^{-7}	0.00×10^{0}
Mosek	1.29	10	5.80556×10^{-1}	0.103	3.75×10^{-5}	1.85×10^{-5}	1.88×10^{-5}
Gurobi	0.31	12	5.80556×10^{-1}	0.105	3.77×10^{-5}	1.87×10^{-5}	1.90×10^{-5}
SGD Primal	7.26	242805	5.83301×10^{-1}	0.966	2.12×10^{-4}	2.06×10^{-4}	2.06×10^{-4}

表 3: Subgradient method for the primal problem.

可以看到，次梯度法运行时间很长，迭代次数很多，与精确解的误差较大。由于将接近0的x调整为0的策略，解的稀疏程度较高。

4.2 (d) Proximal gradient method for the primal problem

之前已经计算过$f(x)$的梯度$\nabla f(x) = A^T(Ax - b)$, the proximal gradient 在第k次迭代的更新公式为:

$$\begin{aligned} x^{k+1} &= \text{prox}_{t_k h}(x^k - t_k \nabla f(x^k)) \\ &= \underset{x \in \mathbb{R}^{n \times l}}{\text{argmin}} \left\{ t_k h(x) + \frac{1}{2}\|x - (x^k - t_k \nabla f(x^k))\|_F^2 \right\} \\ &= \underset{x \in \mathbb{R}^{n \times l}}{\text{argmin}} \left\{ t_k \mu \|x\|_{1,2} + \frac{1}{2}\|x - (x^k - t_k A^T(Ax^k - b))\|_F^2 \right\} \end{aligned}$$

我们可以分别对每一行求最小值。记

$$z_i = (x^k - t_k A^T(Ax^k - b))(i, 1:l), i = 1, \ldots, n$$

那么

$$\begin{aligned} x^{k+1}(i, 1:l) &= \underset{x \in \mathbb{R}^l}{\text{argmin}} \left\{ t_k \mu \|x\|_2 + \frac{1}{2}\|x - z_i\|_2^2 \right\} \\ &= \begin{cases} (\|z_k\|_2 - t_k\mu)\frac{z_k}{\|z_k\|_2}, if \ \ \|z_k\|_2 > t_k\mu \\ 0, if \ \ \|z_k\| \le t_k\mu \end{cases} \end{aligned}$$

代码实现的核心步骤为

```
r = A * x - b;
g = A' * r;
x = prox(xp - t * g, t * mu);
```

Prox算子的实现为

```
function y = prox(x, mu)
    nr = norms(x,2,2);
    y = x .* repmat(max(nr - mu,0)./max(nr, 1e-8),1,size(x,2));
end
```

在数值实验中，我们使用线搜索和BB步长，最终实验结果为

Method	CPU (s)	Iterations	Optimal Value	Sparsity	Err. to Exact	Err. to CVX-Mosek	Err. to CVX-Gurobi
CVX-Mosek	2.66	11	5.80563×10^{-1}	0.103	1.90×10^{-5}	0.00×10^{0}	4.74×10^{-7}
CVX-Gurobi	0.72	25	5.80563×10^{-1}	0.103	1.87×10^{-5}	4.74×10^{-7}	0.00×10^{0}
Mosek	1.29	10	5.80556×10^{-1}	0.103	3.75×10^{-5}	1.85×10^{-5}	1.88×10^{-5}
Gurobi	0.31	12	5.80556×10^{-1}	0.105	3.77×10^{-5}	1.87×10^{-5}	1.90×10^{-5}
SGD Primal	7.26	242805	5.83301×10^{-1}	0.966	2.12×10^{-4}	2.06×10^{-4}	2.06×10^{-4}
ProxGD Primal	0.87	18465	5.80556×10^{-1}	0.103	3.74×10^{-5}	1.84×10^{-5}	1.88×10^{-5}

表 4: Proximal gradient method for the primal problem.

可以看到，Proximal gradient method相比于Subgradient method效果好了很多。

4.3 (e) Fast proximal gradient method for the primal problem

在迭代时进行加速，具体公式如下

$$y^k = x^{k-1} + \frac{k-2}{k+1}(x^{k-1} - x^{k-2})$$
$$x^k = \mathrm{prox}_{t_k h}(y^k - t_k \nabla f(y^k))$$

代码实现的核心步骤为

```
y = x + (k-1)/(k+2)*(x - xp);
x = prox(y - t * g, t * mu);
```

在数值实验中，我们使用线搜索，最终实验结果为

Method	CPU (s)	Iterations	Optimal Value	Sparsity	Err. to Exact	Err. to CVX-Mosek	Err. to CVX-Gurobi
CVX-Mosek	2.66	11	5.80563×10^{-1}	0.103	1.90×10^{-5}	0.00×10^{0}	4.74×10^{-7}
CVX-Gurobi	0.72	25	5.80563×10^{-1}	0.103	1.87×10^{-5}	4.74×10^{-7}	0.00×10^{0}
Mosek	1.29	10	5.80556×10^{-1}	0.103	3.75×10^{-5}	1.85×10^{-5}	1.88×10^{-5}
Gurobi	0.31	12	5.80556×10^{-1}	0.105	3.77×10^{-5}	1.87×10^{-5}	1.90×10^{-5}
SGD Primal	7.26	242805	5.83301×10^{-1}	0.966	2.12×10^{-4}	2.06×10^{-4}	2.06×10^{-4}
ProxGD Primal	0.87	18465	5.80556×10^{-1}	0.103	3.74×10^{-5}	1.84×10^{-5}	1.88×10^{-5}
FProxGD Primal	0.29	6628	5.80556×10^{-1}	0.103	3.71×10^{-5}	1.81×10^{-5}	1.85×10^{-5}

表 5: Fast proximal gradient method for the primal problem.

可以看到，简单的加速策略就使得算法的运行时间大大减少，迭代次数也减少了很多。

4.4 (f) Augmented Lagrangian method for the dual problem

首先，我们推导对偶问题。

对于原问题

$$\min_{x\in\mathbb{R}^{n\times l}} \frac{1}{2}\|Ax-b\|_F^2+\mu\|x\|_{1,2}$$

引入变量 $s=Ax-b\in\mathbb{R}^{m\times l}$，则原问题即为

$$\min_{x\in\mathbb{R}^{n\times l},s\in\mathbb{R}^{m\times l}} \frac{1}{2}\|s\|_F^2+\mu\|x\|_{1,2}$$
$$s.t.\quad Ax-b=s$$

拉格朗日函数：

$$L(x,s,y)=\frac{1}{2}\|s\|_F^2+\mu\|x\|_{1,2}+\langle y,Ax-s-b\rangle$$

其中 $y\in\mathbb{R}^{m\times l}$ 是与约束条件相关的拉格朗日乘子矩阵，$\langle\cdot,\cdot\rangle$ 表示矩阵内积，即 $\langle A,B\rangle=\text{trace}(A^TB)$。

我们的目标是最小化 $L(x,s,y)$ 关于原始变量 x 和 s，然后关于 y 最大化。首先，我们关于 x 和 s 最小化 $L(x,s,y)$。对于 s，我们有：

$$\frac{\partial L}{\partial s}=s-y$$

令 $\frac{\partial L}{\partial s}=0$ 得到 $s=y$。对于 x，我们需要解决以下问题：

$$\min_x \mu\|x\|_{1,2}+\langle A^Ty,x\rangle$$

因此，可以逐行求解。对于每一行 $i=1,\dots,n$，记 $x(i,1:l)=x_i$， $A^Ty(i,1:l)=a_i$，则有

$$\min_{x_i}\mu\left\|x_i\right\|_2+\langle a_i,x_i\rangle=\begin{cases}0,\text{if }\left\|a_i\right\|_2\le\mu\\ -\infty,\text{if }\left\|a_i\right\|_2>\mu\end{cases}$$

因此，对偶问题为

$$\max_{y\in\mathbb{R}^{m\times l}} \quad -\frac{1}{2}\left\|y\right\|_F^2-\langle y,b\rangle$$
$$\text{s.t.}\quad \left\|A^Ty\right\|_{\infty,2}\le\mu$$

也就是如下求最小问题的相反数

$$\begin{aligned}\min_{y\in\mathbb{R}^{m\times l},w\in\mathbb{R}^{n\times l}} \quad & \frac{1}{2}\|y\|_F^2+\langle y,b\rangle \\ \text{s.t.} \quad & A^Ty+z=0 \\ & \|z\|_{\infty,2}\le\mu\end{aligned}$$

拉格朗日函数为：

$$L(y,z,x)=\frac{1}{2}\|y\|_F^2+\langle y,b\rangle-\langle x,A^Ty+z\rangle$$

增广拉格朗日函数为：

$$\begin{aligned}L_\sigma(y,z,x)=&\frac{1}{2}\|y\|_F^2+\langle y,b\rangle-\langle x,A^Ty+z\rangle+\frac{\sigma}{2}\left\|A^Ty+z\right\|_F^2\\ =&\frac{1}{2}\|y\|_F^2+\langle y,b\rangle+\frac{\sigma}{2}\left\|A^Ty+z-\frac{1}{\sigma}x\right\|_F^2-\frac{1}{2\sigma}\|x\|_F^2\end{aligned}$$

因此，迭代公式为：

$$\begin{aligned}y^{k+1},z^{k+1}=&\operatorname*{argmin}_{y,z}\left\{\frac{1}{2}\|y\|_F^2+\langle y,b\rangle+\frac{\sigma}{2}\left\|A^Ty+z-\frac{1}{\sigma}x^k\right\|_F^2\right\}\\ x^{k+1}=&x^k-\sigma\left(A^Ty^{k+1}+z^{k+1}\right)\end{aligned}$$

其中y^{k+1},z^{k+1}难以直接求出，我们在内循环中使用多次投影梯度下降进行迭代求解：

$$\begin{aligned}z^{i+1}=&\mathrm{Proj}_{\{z:\|z\|_{\infty,2}\le\mu\}}\left(z^i-\alpha\sigma\left(A^Ty^i+z^i-\frac{1}{\sigma}x^k\right)\right)\\ y^{i+1}=&y^i-\alpha\left(y^i+b+\sigma A\left(A^Ty^i+z^{i+1}-\frac{1}{\sigma}x^k\right)\right)\end{aligned}$$

代码实现的核心步骤为

```
for i = 1:10
    AtY = A' * y;
    z = proj(z - alpha * sm * (AtY + z - x / sm), mu);
    y = y - alpha * (y + b + sm * A * (AtY + z - x / sm));
end
c = z + A' * y;
x = x - opts.gamma * sm * c;
```

数值实验结果如下

Method	CPU (s)	Iterations	Optimal Value	Sparsity	Err. to Exact	Err. to CVX-Mosek	Err. to CVX-Gurobi
CVX-Mosek	2.66	11	5.80563×10^{-1}	0.103	1.90×10^{-5}	0.00×10^{0}	4.74×10^{-7}
CVX-Gurobi	0.72	25	5.80563×10^{-1}	0.103	1.87×10^{-5}	4.74×10^{-7}	0.00×10^{0}
Mosek	1.29	10	5.80556×10^{-1}	0.103	3.75×10^{-5}	1.85×10^{-5}	1.88×10^{-5}
Gurobi	0.31	12	5.80556×10^{-1}	0.105	3.77×10^{-5}	1.87×10^{-5}	1.90×10^{-5}
SGD Primal	7.26	242805	5.83301×10^{-1}	0.966	2.12×10^{-4}	2.06×10^{-4}	2.06×10^{-4}
ProxGD Primal	0.87	18465	5.80556×10^{-1}	0.103	3.74×10^{-5}	1.84×10^{-5}	1.88×10^{-5}
FProxGD Primal	0.29	6628	5.80556×10^{-1}	0.103	3.71×10^{-5}	1.81×10^{-5}	1.85×10^{-5}
ALM Dual	4.71	1423	5.81258×10^{-1}	0.987	5.31×10^{-4}	5.37×10^{-4}	5.36×10^{-4}

表 6: Augmented Lagrangian method for the dual problem.

可以看到，可能由于求解子问题效率一般，整体运行时间较长，但是迭代次数较少，解的稀疏程度很高。

4.5 (g) Alternating direction method of multipliers for the dual problem

在上一节的推导上，我们可以使用交替方向乘子法(ADMM)来化解子问题。根据增广拉格朗日函数：

$$\begin{aligned} L_\sigma(y,z,x) =& \frac{1}{2}\|y\|_F^2 + \langle y, b\rangle - \langle x, A^T y + z\rangle + \frac{\sigma}{2}\left\|A^T y + z\right\|_F^2 \\ =& \frac{1}{2}\|y\|_F^2 + \langle y, b\rangle + \frac{\sigma}{2}\left\|A^T y + z - \frac{1}{\sigma}x\right\|_F^2 - \frac{1}{2\sigma}\|x\|_F^2 \end{aligned}$$

对z和y依次求最小值，得到迭代公式

$$\begin{aligned} z^{k+1} =& \mathrm{Proj}_{\{z:\|z\|_{\infty,2}\le\mu\}}\left(\frac{1}{\sigma}x^k - A^T y^k\right) \\ y^{k+1} =& \left(I + \sigma AA^T\right)^{-1}\left(Ax^k - b - \sigma A z^{k+1}\right) \\ x^{k+1} =& x^k - \sigma\left(A^T y^{k+1} + z^{k+1}\right) \end{aligned}$$

代码实现的核心步骤为

```
z = proj( - A' * y + x / sm, mu);
h = A * (- z*sm + x) - b;
y = R \ (R' \ h);
```

```
c = z + A' * y;
x = x - opts.gamma * sm * c;
```

数值实验结果如下

Method	CPU (s)	Iterations	Optimal Value	Sparsity	Err. to Exact	Err. to CVX-Mosek	Err. to CVX-Gurobi
CVX-Mosek	2.66	11	5.80563×10^{-1}	0.103	1.90×10^{-5}	0.00×10^{0}	4.74×10^{-7}
CVX-Gurobi	0.72	25	5.80563×10^{-1}	0.103	1.87×10^{-5}	4.74×10^{-7}	0.00×10^{0}
Mosek	1.29	10	5.80556×10^{-1}	0.103	3.75×10^{-5}	1.85×10^{-5}	1.88×10^{-5}
Gurobi	0.31	12	5.80556×10^{-1}	0.105	3.77×10^{-5}	1.87×10^{-5}	1.90×10^{-5}
SGD Primal	7.26	242805	5.83301×10^{-1}	0.966	2.12×10^{-4}	2.06×10^{-4}	2.06×10^{-4}
ProxGD Primal	0.87	18465	5.80556×10^{-1}	0.103	3.74×10^{-5}	1.84×10^{-5}	1.88×10^{-5}
FProxGD Primal	0.29	6628	5.80556×10^{-1}	0.103	3.71×10^{-5}	1.81×10^{-5}	1.85×10^{-5}
ALM Dual	4.71	1423	5.81258×10^{-1}	0.987	5.31×10^{-4}	5.37×10^{-4}	5.36×10^{-4}
ADMM Dual	0.19	320	5.80633×10^{-1}	0.848	6.66×10^{-5}	5.27×10^{-5}	5.29×10^{-5}

表 7: Alternating direction method of multipliers for the dual problem.

可以看到，使用ADMM求解对偶问题是目前效率最高的方法，运行时间和迭代次数都很少，解的稀疏程度也很高。这得益于每次迭代都有解析解。

4.6 (h) Alternating direction method of multipliers with linearization for the primal problem

在Primal问题中

$$\min_{x \in \mathbb{R}^{n \times l}} \frac{1}{2}\|Ax - b\|_F^2 + \mu \|x\|_{1,2}$$

进行拆分，令$z = x$，那么原问题可以改写成

$$\begin{aligned} \min_{x \in \mathbb{R}^{n \times l}, z \in \mathbb{R}^{n \times l}} \quad & \frac{1}{2}\|Ax - b\|_F^2 + \mu \|z\|_{1,2} \\ \text{s.t.} \quad & x = z \end{aligned}$$

引入拉格朗日乘子$y \in \mathbb{R}^{n \times l}$，拉格朗日函数为

$$L(x, z, y) = \frac{1}{2}\|Ax - b\|_F^2 + \mu \|z\|_{1,2} + \langle y, x - z \rangle$$

增广拉格朗日函数为

$$L_\sigma(x, z, y) = \frac{1}{2}\|Ax - b\|_F^2 + \mu \|z\|_{1,2} + \langle y, x - z \rangle + \frac{\sigma}{2}\|x - z\|_F^2$$

使用线性化的ADMM进行迭代，得到迭代公式

$$
\begin{aligned}
x^{k+1} =& x^k - \alpha \left(A^T(Ax^k - b) + y^k + \sigma(x^k - z^k) \right) \\
z^{k+1} =& \mathrm{Prox}_{\alpha\mu\|\cdot\|_{1,2}} \left(z^k + \alpha \left(y^k + \sigma(x^{k+1} - z^k) \right) \right) \\
y^{k+1} =& y^k + \sigma \left(x^{k+1} - z^{k+1} \right)
\end{aligned}
$$

代码实现的核心步骤为

```
x = x - alpha * (A' * (A * x - b) + y + sm * (x - z));
z = prox(z + alpha2 * (y + sm * (x - z)), alpha2 * mu);
y = y + opts.gamma * sm * (x - z);
```

数值实验结果如下

Method	CPU (s)	Iterations	Optimal Value	Sparsity	Err. to Exact	Err. to CVX-Mosek	Err. to CVX-Gurobi
CVX-Mosek	2.66	11	5.80563×10^{-1}	0.103	1.90×10^{-5}	0.00×10^{0}	4.74×10^{-7}
CVX-Gurobi	0.72	25	5.80563×10^{-1}	0.103	1.87×10^{-5}	4.74×10^{-7}	0.00×10^{0}
Mosek	1.29	10	5.80556×10^{-1}	0.103	3.75×10^{-5}	1.85×10^{-5}	1.88×10^{-5}
Gurobi	0.31	12	5.80556×10^{-1}	0.105	3.77×10^{-5}	1.87×10^{-5}	1.90×10^{-5}
SGD Primal	7.26	242805	5.83301×10^{-1}	0.966	2.12×10^{-4}	2.06×10^{-4}	2.06×10^{-4}
ProxGD Primal	0.87	18465	5.80556×10^{-1}	0.103	3.74×10^{-5}	1.84×10^{-5}	1.88×10^{-5}
FProxGD Primal	0.29	6628	5.80556×10^{-1}	0.103	3.71×10^{-5}	1.81×10^{-5}	1.85×10^{-5}
ALM Dual	4.71	1423	5.81258×10^{-1}	0.987	5.31×10^{-4}	5.37×10^{-4}	5.36×10^{-4}
ADMM Dual	0.19	320	5.80633×10^{-1}	0.848	6.66×10^{-5}	5.27×10^{-5}	5.29×10^{-5}
ADMM Primal	23.35	174084	5.80635×10^{-1}	0.872	1.06×10^{-4}	9.32×10^{-5}	9.34×10^{-5}

表 8: Alternating direction method of multipliers with linearization for the primal problem.

可以看到，求解原问题并不适合线性化，因为两个子问题都有显式解。经过测试，如果至少保留一个子问题不用线性化而选择直接求解，那么效率会有提升。反之，如果出于教学需要将两个子问题都线性化，不仅效率低，调参也很困难。

5 Conclusion

Method	CPU (s)	Iterations	Optimal Value	Sparsity	Err. to Exact	Err. to CVX-Mosek	Err. to CVX-Gurobi
CVX-Mosek	2.66	11	5.80563×10^{-1}	0.103	1.90×10^{-5}	0.00×10^{0}	4.74×10^{-7}
CVX-Gurobi	0.72	25	5.80563×10^{-1}	0.103	1.87×10^{-5}	4.74×10^{-7}	0.00×10^{0}
Mosek	1.29	10	5.80556×10^{-1}	0.103	3.75×10^{-5}	1.85×10^{-5}	1.88×10^{-5}
Gurobi	0.31	12	5.80556×10^{-1}	0.105	3.77×10^{-5}	1.87×10^{-5}	1.90×10^{-5}
SGD Primal	7.26	242805	5.83301×10^{-1}	0.966	2.12×10^{-4}	2.06×10^{-4}	2.06×10^{-4}
ProxGD Primal	0.87	18465	5.80556×10^{-1}	0.103	3.74×10^{-5}	1.84×10^{-5}	1.88×10^{-5}
FProxGD Primal	0.29	6628	5.80556×10^{-1}	0.103	3.71×10^{-5}	1.81×10^{-5}	1.85×10^{-5}
ALM Dual	4.71	1423	5.81258×10^{-1}	0.987	5.31×10^{-4}	5.37×10^{-4}	5.36×10^{-4}
ADMM Dual	0.19	320	5.80633×10^{-1}	0.848	6.66×10^{-5}	5.27×10^{-5}	5.29×10^{-5}
ADMM Primal	23.35	174084	5.80635×10^{-1}	0.872	1.06×10^{-4}	9.32×10^{-5}	9.34×10^{-5}

表 9: Comparison of optimization methods

本次数值实验使用的计算机CPU为i5-12500H，一般情况下在单核运行，使用的测试脚本为作业说明中的配套脚本。经过多次测试，发现存在因为轻薄本调度策略不同而导致的运行时间差异，因此在表格中给出的运行时间为最后一次测试的结果，仅供参考。

本次数值实验的参数和策略只是进行了粗略的调整，因此还有提升空间。在所有的方法中，ADMM Dual方法的运行时间最短，迭代次数也最少，解的稀疏程度也很高，超越了Mosek和Gurobi求解器的结果，具有较大的价值。